

Rating

Rating The `<aava-rating>` component provides an intuitive and accessible star rating interface with support for half-star ratings, multiple size variants, and comprehensive keyboard navigation.

Perfect for user feedback, product reviews, and any scenario requiring rating input or display. How to use ■ import { AavaRatingComponent } from "@aava/play-core"; Basic Usage ■ Simple rating implementation with default 5-star scale and interactive functionality. Angular Preview Code `<aava-rating [value] = " ratingValue " (rated) = " onRatingChange($event) " ></ aava-rating >` `onRatingChange (value : number) { console . log ('Rating changed to:' , value) ; }` Sizes ■ Four size variants to accommodate different interface densities and visual hierarchy requirements.

Angular Preview Code `<!-- Different size variants -->` `< aava-rating [value] = " ratingValue " size = " xs " (rated) = " onRatingChange($event) " ></ aava-rating >` `< aava-rating [value] = " ratingValue " size = " sm " (rated) = " onRatingChange($event) " ></ aava-rating >` `< aava-rating [value] = " ratingValue " size = " md " (rated) = " onRatingChange($event) " ></ aava-rating >` `< aava-rating [value] = " ratingValue " size = " lg " (rated) = " onRatingChange($event) " ></ aava-rating >` `< aava-rating [value] = " ratingValue " [size] = " 40 " (rated) = " onRatingChange($event) " ></ aava-rating >` `ratingValue = 3.5 ; ononRatingChange (value : number) { this . ratingValue = value ; console . log ('Rating changed to:' , value) ; }` Available Sizes ■ xs (Extra Small) - 16px stars for very compact interfaces sm (Small) - 20px stars for dense interfaces md (Medium) - 24px stars for standard layouts (default) lg (Large) - 32px stars for prominent placements and better accessibility

Custom - Numeric values for precise sizing requirements Half-Star Ratings ■ Support for precise half-star ratings (e.g., 4.5 stars) with intuitive click positioning. Angular Preview Code `<!-- Half-star ratings -->` `< aava-rating [value] = " 3.5 " (rated) = " onRatingChange($event) " ></ aava-rating >` `< aava-rating [value] = " 4.5 " (rated) = " onRatingChange($event) " ></ aava-rating >` `< aava-rating [value] = " 2.5 " (rated) = " onRatingChange($event) " ></ aava-rating >` `<!-- Interactive half-star selection -->` `< aava-rating [value] = " currentRating " (rated) = " ratingChange($event) " ></ aava-rating >` `currentRating = 0 ; onRatingChange (rating : number) { this . currentRating = rating ; console . log ('Rating changed to:' , rating) ; }` Half-Star Features ■ Click Positioning - Left half of star = half rating, right half = full rating Hover Preview - Visual feedback shows potential rating before clicking Precise Control - Support for ratings like 3.5, 4.5, etc. Intuitive UX - Natural

interaction pattern users expect Readonly Mode ■ Display-only mode for showing existing ratings without user interaction. Angular Preview Code `< aava-rating [value] = " 4.5 " [readonly] = " true " ></ aava-rating >` Readonly Features ■ Non-interactive - No click or hover effects Display Only - Perfect for showing existing ratings Accessibility - Maintains proper ARIA attributes Consistent Styling - Same visual appearance as interactive mode Show Value ■ Display the numeric rating value alongside the visual stars. Angular Preview Code Value Display Features ■ Numeric Rating - Shows exact rating (e.g., "4.5") Size Variants - Value text scales with star size Positioning - Value appears to the right of stars Formatting - Always shows one decimal place for precision

Custom Maximum ■ Flexible rating scales beyond the default 5-star system. Angular Preview
Code Custom Scale Features ■ Flexible Range - Support for 3, 4, 5, 10, or any number of stars
Consistent Behavior - Same interaction patterns regardless of scale Half-Star Support - Works
with any maximum value Accessibility - Proper ARIA attributes for custom scales Accessibility ■
Accessibility Features ■ Keyboard Navigation - Full keyboard support with arrow keys ARIA
Compliance - Proper role="radiogroup" and aria-checked attributes Screen Reader Support -
Clear announcements of current rating Focus Management - Visible focus indicators for keyboard
users High Contrast - Enhanced visibility in high contrast modes Motion Preferences - Respects
user's reduced motion settings Keyboard Shortcuts ■ Arrow Right/Up - Increase rating by 1 star
Arrow Left/Down - Decrease rating by 1 star Enter/Space - Select the currently focused star
Tab/Shift+Tab - Navigate between stars API Reference ■ Inputs ■ Property Type Default
Description value number 0 Current rating value (supports halves like 4.5) max number 5
Maximum number of stars in the rating scale readonly boolean false Whether the rating is
read-only (non-interactive) size number | 'xs' | 'sm' | 'md' | 'lg' | 'md' Size of the stars (predefined or
custom pixel values) showValue boolean false Whether to display the numeric rating value
Outputs ■ Event Type Description rated EventEmitter<number> Emitted when user changes the
rating value CSS Custom Properties ■ Property Description --rating-label-font-family Font family
for rating label text --rating-label-font-weight Font weight for rating label text --rating-label-font-size
Font size (used as line-height) for rating label text --rating-label-color Text color for rating label
--rating-label-letter-spacing-sm Letter spacing for small label text
--rating-label-letter-spacing-medium Letter spacing for medium label text
--rating-label-letter-spacing-lg Letter spacing for large label text --rating-value-font-size-sm Font
size for small value variants (xs & sm) --rating-value-font-size-md Font size for medium value
variant --rating-value-font-size-lg Font size for large value variant Best Practices ■ Design
Guidelines ■ Choose appropriate sizes - Use larger sizes for primary rating displays, smaller for
secondary Consider half-star support - Enable for precise rating needs, disable for simpler
interfaces Show value when needed - Display numeric ratings for clarity in review systems Use
consistent scales - Stick to common scales (5-star, 10-star) for user familiarity Position
strategically - Place ratings near relevant content for context Accessibility ■ Always provide labels
- Use descriptive labels for screen reader context Test keyboard navigation - Ensure full keyboard
accessibility Consider motion preferences - Respect user's reduced motion settings Maintain
contrast - Ensure sufficient contrast for all star states Provide alternatives - Consider text-based
rating alternatives for complex cases Performance ■ Optimize re-renders - Use OnPush change
detection strategy when possible Efficient event handling - Optimize mouse and keyboard event
handlers Image optimization - Use optimized SVG assets for stars Memory management - Clean
up event listeners properly Form Integration ■ Angular Forms - Integrate with reactive and
template-driven forms Validation - Implement appropriate validation for rating inputs Default
values - Provide sensible defaults for new ratings Error handling - Handle edge cases and invalid
inputs gracefully Use Cases ■ Product Reviews - E-commerce product rating systems Service
Feedback - Customer satisfaction ratings Content Rating - Movie, book, or content ratings Skill

Assessment - Employee or skill evaluation systems Quality Metrics - Internal quality or performance ratings Technical Notes ■ Star Asset Requirements ■ The component expects three SVG assets: star-filled.svg - For fully rated stars star-half.svg - For half-rated stars star-outline.svg - For empty stars Half-Star Logic ■ Half-star ratings are determined by click position: Left half of star = index + 0.5 Right half of star = index + 1.0 Size Mapping ■ Predefined sizes map to pixel values: extra small : 16px small : 20px medium : 24px (default) large : 32px Custom numeric values are used directly for precise sizing requirements. Event Handling ■ The component handles multiple interaction types: Mouse : Click for selection, hover for preview Keyboard : Arrow navigation, Enter/Space for selection Touch : Click events work on touch devices Programmatic : Direct value changes via input binding

```
<aava-rating [value]="ratingValue" (rated)="onRatingChange($event)">
</aava-rating>
```

```
onRatingChange(value: number) {
  console.log('Rating changed to:', value);
}
```

```
<!-- Different size variants -->
<aava-rating [value]="ratingValue" size="xs" (rated)="onRatingChange($event)">
</aava-rating>
<aava-rating [value]="ratingValue" size="sm" (rated)="onRatingChange($event)">
</aava-rating>
<aava-rating [value]="ratingValue" size="md" (rated)="onRatingChange($event)">
</aava-rating>
<aava-rating [value]="ratingValue" size="lg" (rated)="onRatingChange($event)">
</aava-rating>
<aava-rating [value]="ratingValue" [size]="40" (rated)="onRatingChange($event)">
</aava-rating>
```

```
ratingValue = 3.5;

ononRatingChange(value: number) {
  this.ratingValue = value;
  console.log('Rating changed to:', value);
}
```

```
<!-- Half-star ratings -->
<aava-rating [value]="3.5" (rated)="onRatingChange($event)"></aava-rating>
<aava-rating [value]="4.5" (rated)="onRatingChange($event)"></aava-rating>
<aava-rating [value]="2.5" (rated)="onRatingChange($event)"></aava-rating>
<!-- Interactive half-star selection -->
<aava-rating
  [value]="currentRating"
  (rated)="ratingChange($event)"
></aava-rating>
```

```
currentRating = 0;
onRatingChange(rating: number) {
  this.currentRating = rating;
  console.log('Rating changed to:', rating);
}
```

```
<aava-rating [value]="4.5" [readonly]="true"></aava-rating>
```

■ No code found

■ No code found